

ADOM - Application-centric Data Object Management

Generated by Doxygen 1.9.1

1 Todo List	1
2 Namespace Index	3
2.1 Namespace List	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Namespace Documentation	9
5.1 adom Namespace Reference	9
5.1.1 Function Documentation	9
5.1.1.1 init_console_log()	9
5.1.1.2 init_file_log()	9
6 Class Documentation	11
6.1 AdomMessage Class Reference	11
6.1.1 Detailed Description	11
6.1.2 Constructor & Destructor Documentation	11
6.1.2.1 AdomMessage() [1/2]	12
6.1.2.2 AdomMessage() [2/2]	12
6.1.3 Member Function Documentation	12
6.1.3.1 adomMessageToNet()	12
6.1.3.2 clear()	12
6.1.3.3 netToAdomMessage()	12
6.1.4 Member Data Documentation	13
6.1.4.1 adom_message_length_	13
6.1.4.2 payload_	13
6.1.4.3 payload_length_	13
6.1.4.4 type_	13
6.2 ApplicationInterface Class Reference	13
6.2.1 Constructor & Destructor Documentation	14
6.2.1.1 ApplicationInterface()	14
6.2.2 Member Function Documentation	14
6.2.2.1 initialize()	14
6.2.2.2 readPartialData() [1/2]	15
6.2.2.3 readPartialData() [2/2]	15
6.2.2.4 registerNewData()	16
6.2.2.5 registerNewTopic()	16
6.2.2.6 stop()	16
6.3 AppRequest Struct Reference	17
6.3.1 Member Data Documentation	17

6.3.1.1 access_end_address	17
6.3.1.2 access_start_address	17
6.3.1.3 object_address	17
6.3.1.4 sequence_nr	17
6.3.1.5 topic	17
6.4 DataAnnouncement Class Reference	18
6.4.1 Detailed Description	18
6.4.2 Constructor & Destructor Documentation	18
6.4.2.1 DataAnnouncement() [1/3]	18
6.4.2.2 DataAnnouncement() [2/3]	19
6.4.2.3 DataAnnouncement() [3/3]	19
6.4.3 Member Function Documentation	19
6.4.3.1 clear()	19
6.4.3.2 dataAnnouncementToNet()	19
6.4.3.3 netToDataAnnouncement()	19
6.4.3.4 print()	20
6.4.4 Member Data Documentation	20
6.4.4.1 associated_object_	20
6.4.4.2 home_directory_	20
6.4.4.3 message_length_	20
6.4.4.4 structure_	20
6.5 DataBlockHeader Struct Reference	21
6.5.1 Detailed Description	21
6.5.2 Constructor & Destructor Documentation	21
6.5.2.1 DataBlockHeader() [1/2]	21
6.5.2.2 DataBlockHeader() [2/2]	21
6.5.3 Member Data Documentation	21
6.5.3.1 associated_object	22
6.5.3.2 block_id	22
6.5.3.3 block_size	22
6.5.3.4 data_address_offset	22
6.6 DataObjectInstance Struct Reference	22
6.6.1 Detailed Description	22
6.6.2 Constructor & Destructor Documentation	23
6.6.2.1 DataObjectInstance()	23
6.6.3 Member Data Documentation	23
6.6.3.1 object_sequence_nr	23
6.6.3.2 object_type	23
6.7 DataRequest Class Reference	23
6.7.1 Detailed Description	24
6.7.2 Constructor & Destructor Documentation	24
6.7.2.1 DataRequest() [1/2]	24

6.7.2.2 DataRequest() [2/2]	24
6.7.3 Member Function Documentation	24
6.7.3.1 clear()	24
6.7.3.2 dataRequestToNet()	24
6.7.3.3 netToDataRequest()	25
6.7.3.4 print()	25
6.7.4 Member Data Documentation	25
6.7.4.1 home_directory_	25
6.7.4.2 message_length_	25
6.7.4.3 object_block_count_	25
6.7.4.4 object_block_validity_	26
6.7.4.5 object_instance_	26
6.7.4.6 reader_application_	26
6.7.4.7 request_id_	26
6.8 DataTransport Class Reference	26
6.8.1 Detailed Description	27
6.8.2 Constructor & Destructor Documentation	27
6.8.2.1 DataTransport() [1/2]	27
6.8.2.2 DataTransport() [2/2]	27
6.8.3 Member Function Documentation	27
6.8.3.1 clear()	27
6.8.3.2 dataTransportToNet()	27
6.8.3.3 netToDataTransport()	28
6.8.3.4 print()	28
6.8.4 Member Data Documentation	28
6.8.4.1 associated_object_	28
6.8.4.2 associated_request_id_	28
6.8.4.3 block_id_	28
6.8.4.4 block_size_	29
6.8.4.5 data_	29
6.8.4.6 message_length_	29
6.9 Directory Class Reference	29
6.9.1 Constructor & Destructor Documentation	30
6.9.1.1 Directory()	30
6.9.2 Member Function Documentation	31
6.9.2.1 addNewObjectSample()	31
6.9.2.2 checkAvailability()	31
6.9.2.3 freeAllObjects()	31
6.9.2.4 getLastObjectSample()	32
6.9.2.5 getLatestSequencenumberOfTrackedTopic()	33
6.9.2.6 getNumberOfTrackedObjects()	33
6.9.2.7 getSample()	33

6.9.2.8 initiate()	34
6.9.2.9 join()	34
6.9.2.10 printTrackedObjectSamples()	34
6.9.2.11 sendDataRequest()	34
6.9.2.12 setRequestedBlocks()	35
6.9.2.13 stop()	35
6.9.2.14 terminate()	36
6.9.3 Member Data Documentation	36
6.9.3.1 other_directory_descriptor_	36
6.9.3.2 request_tracking_	36
6.9.3.3 request_tracking_lock_	36
6.10 EntityDescriptor Struct Reference	36
6.10.1 Detailed Description	37
6.10.2 Constructor & Destructor Documentation	37
6.10.2.1 EntityDescriptor()	37
6.10.3 Member Data Documentation	37
6.10.3.1 entity_id	37
6.10.3.2 ingress_port	37
6.10.3.3 ip_address	37
6.11 ObjectSample Class Reference	38
6.11.1 Detailed Description	38
6.11.2 Constructor & Destructor Documentation	38
6.11.2.1 ObjectSample()	38
6.11.3 Member Function Documentation	38
6.11.3.1 getBlockCount()	39
6.11.3.2 getHeader() [1/2]	39
6.11.3.3 getHeader() [2/2]	39
6.11.3.4 getHomeDirectory()	39
6.11.3.5 getSequenceNumber()	39
6.11.3.6 getStructure()	39
6.11.3.7 getTopic()	39
6.11.3.8 isAvailable()	40
6.11.3.9 makeAvailable()	40
6.11.3.10 printObjectSample()	40
6.11.4 Member Data Documentation	40
6.11.4.1 object_sample_data_	40
6.12 RequestTracking Struct Reference	40
6.12.1 Member Data Documentation	40
6.12.1.1 cv	41
6.12.1.2 received_blocks	41
6.12.1.3 requested_blocks	41
6.13 SafeQueue< T > Class Template Reference	41

6.13.1 Detailed Description	42
6.13.2 Constructor & Destructor Documentation	42
6.13.2.1 SafeQueue()	42
6.13.2.2 ~SafeQueue()	42
6.13.3 Member Function Documentation	42
6.13.3.1 dequeue() [1/3]	43
6.13.3.2 dequeue() [2/3]	43
6.13.3.3 dequeue() [3/3]	43
6.13.3.4 empty()	43
6.13.3.5 enqueue()	44
6.13.4 Member Data Documentation	44
6.13.4.1 queue_event	44
6.13.4.2 queue_lock	44
6.13.4.3 safe_queue	44
6.14 socket_endpoint Struct Reference	45
6.14.1 Detailed Description	45
6.14.2 Constructor & Destructor Documentation	45
6.14.2.1 socket_endpoint() [1/2]	45
6.14.2.2 socket_endpoint() [2/2]	45
6.14.3 Member Data Documentation	45
6.14.3.1 ip_addr	45
6.14.3.2 port	46
6.15 Structure Struct Reference	46
6.15.1 Detailed Description	46
6.15.2 Constructor & Destructor Documentation	46
6.15.2.1 Structure() [1/2]	47
6.15.2.2 Structure() [2/2]	47
6.15.3 Member Data Documentation	47
6.15.3.1 block_cols	47
6.15.3.2 block_rows	47
6.15.3.3 object_channels	47
6.15.3.4 object_height	47
6.15.3.5 object_width	47
6.15.3.6 type	47
7 File Documentation	49
7.1 include/adom/application_interface.hpp File Reference	49
7.2 include/adom/directory.hpp File Reference	49
7.3 include/adom/log.hpp File Reference	50
7.3.1 Macro Definition Documentation	51
7.3.1.1 AppLog	51
7.3.1.2 logDebug	51

7.3.1.3 logError	51
7.3.1.4 logFatal	51
7.3.1.5 logInfo	51
7.3.1.6 logTrace	51
7.3.1.7 logWarning	52
7.4 include/adom/opencv_helper.h File Reference	52
7.4.1 Function Documentation	52
7.4.1.1 findBlock()	52
7.4.1.2 findBlockFromAddress()	53
7.4.1.3 findBlocksForReadAccess()	53
7.5 include/adom/parameters.h File Reference	54
7.5.1 Macro Definition Documentation	55
7.5.1.1 MAX_COUNT_MANAGED_OBJECTS_PER_TOPIC	55
7.5.1.2 PRINTF_BINARY_PATTERN_INT8	55
7.5.1.3 PRINTF_BYTE_TO_BINARY_INT8	55
7.5.1.4 PROTOCOL_TEST	55
7.5.2 Enumeration Type Documentation	55
7.5.2.1 Topic	55
7.5.3 Variable Documentation	56
7.5.3.1 adom_message_overhead	56
7.5.3.2 CHANNEL_NUMBER	56
7.5.3.3 EMPTY_QUEUE_ERROR	56
7.5.3.4 FULL_HD_BLOCK_ROWS	56
7.5.3.5 FULL_HD_BLOCKS_IN_ROW	56
7.5.3.6 FULL_HD_H_PIXEL_PER_IMAGE	56
7.5.3.7 FULL_HD_V_PIXEL_PER_IMAGE	57
7.5.3.8 H_PIXEL_PER_BLOCK	57
7.5.3.9 HD_BLOCK_ROWS	57
7.5.3.10 HD_BLOCKS_IN_ROW	57
7.5.3.11 HD_H_PIXEL_PER_IMAGE	57
7.5.3.12 HD_V_PIXEL_PER_IMAGE	57
7.5.3.13 home_dir_ingress_port	57
7.5.3.14 log_directory	57
7.5.3.15 max_buffer_length	58
7.5.3.16 MAX_NUM_TRACKED_SAMPLES	58
7.5.3.17 publisher_ip	58
7.5.3.18 req_dir_ingress_port	58
7.5.3.19 string_length	58
7.5.3.20 subscriber_ip	58
7.5.3.21 TOTAL_BLOCK_SIZE	58
7.5.3.22 V_PIXEL_PER_BLOCK	58
7.6 include/adom/protocol.hpp File Reference	59

7.6.1 Enumeration Type Documentation	59
7.6.1.1 BlockValidity	59
7.6.1.2 MessageType	60
7.6.2 Function Documentation	60
7.6.2.1 operator==()	60
7.7 include/adom/safe_queue.hpp File Reference	60
7.8 include/adom/socket_endpoint.hpp File Reference	61
7.9 include/adom/translation.h File Reference	61
7.9.1 Enumeration Type Documentation	61
7.9.1.1 StructureType	61
7.9.2 Function Documentation	62
7.9.2.1 translate_from_uchar()	62
7.9.2.2 translate_to_uchar() [1/2]	62
7.9.2.3 translate_to_uchar() [2/2]	62
7.9.2.4 translate_uchar_access()	62
7.10 src/cpp/adom/application_interface.cpp File Reference	63
7.11 src/cpp/adom/directory.cpp File Reference	63
7.12 src/cpp/adom/log.cpp File Reference	63
7.13 src/cpp/adom/protocol.cpp File Reference	63
Index	65

Chapter 1

Todo List

Class `DataObjectInstance`

- o Umbenennen, eher `DataObjectIdentifier`

Member `Directory::other_directory_descriptor_`

- make private

Member `Directory::sendDataRequest` (`EntityDescriptor` home_directory, `std::shared_ptr< ObjectSample >`, `std::vector< uint16_t >` block_ids)

- make private

- make private

Member `translate_from_uchar` (`unsigned char *data`, `Structure` structure)

- o check for allignement issues, is the new data buffer big enough etc

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

adom	9
--------------------------------	---

Chapter 3

Class Index

3.1 Class List

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

AdomMessage	Application Data Object Management Messages are used as Wrapper Data Announcements, for Data Request, and Date Transports	11
ApplicationInterface	13
AppRequest	17
DataAnnouncement	DataAnnouncement is used by a home directory (publisher) to inform a remote (subscribed) directory of a new available sample	18
DataBlockHeader	Struct to describe a Data Block, that is part of an Object instance	21
DataObjectInstance	Struct to describe an Object instance with its sequence number within a topic or object type . .	22
DataRequest	DataRequest is used by a remote (subscribed) directory (reader_application_) to request specific blocks of a sample object managed by the data's (publihser/) home directory (home_directory_) via a block_validity_matrix	23
DataTransport	DataTransport is used by a home directory (publisher) to transport requested blocks of a sample object to a remote (subscribed) directory	26
Directory	29
EntityDescriptor	This struct is utilized to describe an Entity of a directory. This struct must be filled prior to setting up the directory struct itself	36
ObjectSample	Class to decribe an ObjectSample	38
RequestTracking	40
SafeQueue< T >	A thread safe queue	41
socket_endpoint	Describes the parameters of an boost asio endpoint. Used for passing endpoints to functions .	45
Structure	To decompose the object sample into data blocks, that are beneficial to the data type or user application's use of the data, the user should specify the proper structure of the decomposed object sample	46

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

include/adom/application_interface.hpp	49
include/adom/directory.hpp	49
include/adom/log.hpp	50
include/adom/opencv_helper.h	52
include/adom/parameters.h	54
include/adom/protocol.hpp	59
include/adom/safe_queue.hpp	60
include/adom/socket_endpoint.hpp	61
include/adom/translation.h	61
src/cpp/adom/application_interface.cpp	63
src/cpp/adom/directory.cpp	63
src/cpp/adom/log.cpp	63
src/cpp/adom/protocol.cpp	63

Chapter 5

Namespace Documentation

5.1 adom Namespace Reference

Functions

- void [init_file_log](#) (std::string log_prefix, std::string log_suffix)
- void [init_console_log](#) ()

5.1.1 Function Documentation

5.1.1.1 [init_console_log\(\)](#)

```
void adom::init_console_log ( )
```

5.1.1.2 [init_file_log\(\)](#)

```
void adom::init_file_log (
    std::string log_prefix,
    std::string log_suffix )
```


Chapter 6

Class Documentation

6.1 AdomMessage Class Reference

Application Data Object Management Messages are used as Wrapper Data Announcements, for Data Request, and Date Transports.

```
#include <protocol.hpp>
```

Public Member Functions

- [AdomMessage](#) ()
- [AdomMessage](#) ([MessageType](#) type, uint16_t payload_length, char *msg)
- void [adomMessageToNet](#) (char *msg)
This function is called to serialize Application Data Object Management Messages for transport.
- void [netToAdomMessage](#) (char *msg)
This function is called to deserialize received Application Data Object Management Messages.
- void [clear](#) ()
This function is called to clear all attributes of the Application Data Object Management Messages for reuse.

Public Attributes

- enum [MessageType](#) type_
- uint16_t [payload_length_](#)
- std::vector< char > [payload_](#)
- uint16_t [adom_message_length_](#)

6.1.1 Detailed Description

Application Data Object Management Messages are used as Wrapper Data Announcements, for Data Request, and Date Transports.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 AdomMessage() [1/2]

```
AdomMessage::AdomMessage ( ) [inline]
```

6.1.2.2 AdomMessage() [2/2]

```
AdomMessage::AdomMessage (
    MessageType type,
    uint16_t payload_length,
    char * msg ) [inline]
```

6.1.3 Member Function Documentation

6.1.3.1 adomMessageToNet()

```
void AdomMessage::adomMessageToNet (
    char * msg )
```

This function is called to serialize Application Data Object Management Messages for transport.

Parameters

<i>msg</i>	Pointer to buffer to write the serialized message into
------------	--

6.1.3.2 clear()

```
void AdomMessage::clear ( )
```

This function is called to clear all attributes of the Application Data Object Management Messages for reuse.

This function is called to deserialize received Application Data Object Management Messages.

Parameters

<i>msg</i>	Pointer to buffer to the serialized message
------------	---

6.1.3.3 netToAdomMessage()

```
void AdomMessage::netToAdomMessage (
```

```
char * msg )
```

This function is called to deserialize received Application Data Object Management Messages.

Parameters

<i>msg</i>	Pointer to buffer to the serialized message
------------	---

6.1.4 Member Data Documentation

6.1.4.1 adom_message_length_

```
uint16_t AdomMessage::adom_message_length_
```

6.1.4.2 payload_

```
std::vector<char> AdomMessage::payload_
```

6.1.4.3 payload_length_

```
uint16_t AdomMessage::payload_length_
```

6.1.4.4 type_

```
enum MessageType AdomMessage::type_
```

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

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

6.2 ApplicationInterface Class Reference

```
#include <application_interface.hpp>
```

Public Member Functions

- [ApplicationInterface](#) ([EntityDescriptor](#) home_directory_descriptor)
Construct a new Application Interface object.
- void [initialize](#) (void)
Read static definition of other directories and topic structures etc.
- void [registerNewTopic](#) ([Topic](#) type, [Structure](#) structure)
Send a Request to the associated [Directory](#) to: Write a new (complete) sample to the managed shared memory (called from a "Publisher" or "Writer")
- void [registerNewData](#) ([Topic](#) type, unsigned char *data)
Send a Request to the associated [Directory](#) to: Write a new (complete) sample (called from a "Publisher" or "Writer")
- std::future< int > [readPartialData](#) ([Topic](#) type, std::vector< uint16_t > associated_blocks, unsigned char *output_data)
Send a Request to the associated [Directory](#) to: Partially read data from a sample (called from a "Subscriber" or "Reader")
- std::future< int > [readPartialData](#) ([Topic](#) type, int sequence_nr, std::vector< uint16_t > associated_blocks, unsigned char *output_data)
Send a Request to the associated [Directory](#) to: Partially read data from a sample (called from a "Subscriber" or "Reader")
- void [stop](#) ()
Stopping all processes within the interface.

6.2.1 Constructor & Destructor Documentation

6.2.1.1 ApplicationInterface()

```
ApplicationInterface::ApplicationInterface (
    EntityDescriptor home_directory_descriptor ) [inline]
```

Construct a new Application Interface object.

Parameters

<i>associated_directory_descriptor</i>
--

6.2.2 Member Function Documentation

6.2.2.1 initialize()

```
void ApplicationInterface::initialize (
    void )
```

Read static definition of other directories and topic structures etc.

Temporary static definition of other directory

6.2.2.2 readPartialData() [1/2]

```
std::future< int > ApplicationInterface::readPartialData (
    Topic type,
    int sequence_nr,
    std::vector< uint16_t > associated_blocks,
    unsigned char * output_data )
```

Send a Request to the associated [Directory](#) to: Partially read data from a sample (called from a "Subscriber" or "Reader")

Parameters

<i>topic_name</i>	name of the topic, whose object sample is read partially
<i>sequence_nr</i>	sequence number of the sample
<i>associated_blocks</i>	associated_blocks for this read access
<i>output_data</i>	data to write requested block into

6.2.2.3 readPartialData() [2/2]

```
std::future< int > ApplicationInterface::readPartialData (
    Topic type,
    std::vector< uint16_t > associated_blocks,
    unsigned char * output_data )
```

Send a Request to the associated [Directory](#) to: Partially read data from a sample (called from a "Subscriber" or "Reader")

Parameters

<i>type</i>	
<i>associated_blocks</i>	
<i>output_data</i>	

Returns

`std::future<int>`

Parameters

<i>topic_name</i>	name of the topic, whose object sample is read partially
<i>associated_blocks</i>	associated_blocks for this read access
<i>output_data</i>	data to write requested block into

6.2.2.4 registerNewData()

```
void ApplicationInterface::registerNewData (
    Topic type,
    unsigned char * data )
```

Send a Request to the associated [Directory](#) to: Write a new (complete) sample (called from a "Publisher" or "Writer")

Send a Request to the associated [Directory](#) to: Write a new (complete) sample to the managed shared memory (called from a "Publisher" or "Writer")

Parameters

<i>type</i>	name of the topic, whose object sample is written
<i>data</i>	data to be registered

6.2.2.5 registerNewTopic()

```
void ApplicationInterface::registerNewTopic (
    Topic type,
    Structure data_structure )
```

Send a Request to the associated [Directory](#) to: Write a new (complete) sample to the managed shared memory (called from a "Publisher" or "Writer")

Parameters

<i>type</i>	name of the topic, whose object sample is written
<i>structure</i>	structure of the data type required for parsing and specification of blocks
<i>type</i>	name of the topic, whose object sample is written
<i>structure</i>	structure of the data type required for parsing specification of blocks

6.2.2.6 stop()

```
void ApplicationInterface::stop ( )
```

Stopping all processes within the interface.

Stopping all processes within directory.

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

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

6.3 AppRequest Struct Reference

```
#include <protocol.hpp>
```

Public Attributes

- enum [Topic](#) `topic`
- uint16_t `sequence_nr`
- void * `object_address`
- void * `access_start_address`
- void * `access_end_address`

6.3.1 Member Data Documentation

6.3.1.1 `access_end_address`

```
void* AppRequest::access_end_address
```

6.3.1.2 `access_start_address`

```
void* AppRequest::access_start_address
```

6.3.1.3 `object_address`

```
void* AppRequest::object_address
```

6.3.1.4 `sequence_nr`

```
uint16_t AppRequest::sequence_nr
```

6.3.1.5 `topic`

```
enum Topic AppRequest::topic
```

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

- [include/adom/protocol.hpp](#)

6.4 DataAnnouncement Class Reference

[DataAnnouncement](#) is used by a home directory (publisher) to inform a remote (subscribed) directory of a new available sample.

```
#include <protocol.hpp>
```

Public Member Functions

- [DataAnnouncement](#) ()
- [DataAnnouncement](#) ([EntityDescriptor](#) home_directory, [Topic](#) object_type, uint16_t object_sequence_nr)
- [DataAnnouncement](#) ([EntityDescriptor](#) home_directory, [Topic](#) object_type, uint16_t object_sequence_nr, [Structure](#) structure)
- void [dataAnnouncementToNet](#) (char *msg)
This function is called to serialize [DataAnnouncement](#) for transport.
- void [netToDataAnnouncement](#) (char *msg)
This function is called to deserialize a received [DataAnnouncement](#).
- void [clear](#) ()
This function is called to clear all attributes of the [DataAnnouncement](#) for reuse.
- void [print](#) ()
This function prints out the [DataAnnouncement](#).

Public Attributes

- [EntityDescriptor](#) home_directory_
- [DataObjectInstance](#) associated_object_
- [Structure](#) structure_
- uint16_t message_length_

6.4.1 Detailed Description

[DataAnnouncement](#) is used by a home directory (publisher) to inform a remote (subscribed) directory of a new available sample.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 DataAnnouncement() [1/3]

```
DataAnnouncement::DataAnnouncement ( ) [inline]
```

6.4.2.2 DataAnnouncement() [2/3]

```
DataAnnouncement::DataAnnouncement (
    EntityDescriptor home_directory,
    Topic object_type,
    uint16_t object_sequence_nr ) [inline]
```

6.4.2.3 DataAnnouncement() [3/3]

```
DataAnnouncement::DataAnnouncement (
    EntityDescriptor home_directory,
    Topic object_type,
    uint16_t object_sequence_nr,
    Structure structure ) [inline]
```

6.4.3 Member Function Documentation

6.4.3.1 clear()

```
void DataAnnouncement::clear ( )
```

This function is called to clear all attributes of the [DataAnnouncement](#) for reuse.

6.4.3.2 dataAnnouncementToNet()

```
void DataAnnouncement::dataAnnouncementToNet (
    char * msg )
```

This function is called to serialize [DataAnnouncement](#) for transport.

Parameters

<i>msg</i>	Pointer to buffer to write the serialized DataAnnouncement into
------------	---

6.4.3.3 netToDataAnnouncement()

```
void DataAnnouncement::netToDataAnnouncement (
    char * msg )
```

This function is called to deserialize a received [DataAnnouncement](#).

Parameters

<i>msg</i>	Pointer to buffer to the serialized DataAnnouncement
------------	--

6.4.3.4 print()

```
void DataAnnouncement::print ( )
```

This function prints out the [DataAnnouncement](#).

6.4.4 Member Data Documentation

6.4.4.1 associated_object_

```
DataObjectInstance DataAnnouncement::associated_object_
```

6.4.4.2 home_directory_

```
EntityDescriptor DataAnnouncement::home_directory_
```

6.4.4.3 message_length_

```
uint16_t DataAnnouncement::message_length_
```

6.4.4.4 structure_

```
Structure DataAnnouncement::structure_
```

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

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

6.5 DataBlockHeader Struct Reference

Struct to describe a Data Block, that is part of an Object instance.

```
#include <protocol.hpp>
```

Public Member Functions

- [DataBlockHeader](#) ()
- [DataBlockHeader](#) ([Topic](#) topic, uint16_t updated_sequence_nr, uint16_t [block_id](#), uint16_t [block_size](#), long [data_address_offset](#))

Public Attributes

- [DataObjectInstance](#) [associated_object](#)
- uint16_t [block_id](#)
- uint16_t [block_size](#)
- long [data_address_offset](#)

6.5.1 Detailed Description

Struct to describe a Data Block, that is part of an Object instance.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 DataBlockHeader() [1/2]

```
DataBlockHeader::DataBlockHeader ( ) [inline]
```

6.5.2.2 DataBlockHeader() [2/2]

```
DataBlockHeader::DataBlockHeader (
    Topic topic,
    uint16_t updated_sequence_nr,
    uint16_t block_id,
    uint16_t block_size,
    long data_address_offset ) [inline]
```

6.5.3 Member Data Documentation

6.5.3.1 associated_object

`DataObjectInstance` `DataBlockHeader::associated_object`

6.5.3.2 block_id

`uint16_t` `DataBlockHeader::block_id`

6.5.3.3 block_size

`uint16_t` `DataBlockHeader::block_size`

6.5.3.4 data_address_offset

`long` `DataBlockHeader::data_address_offset`

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

- `include/adom/protocol.hpp`

6.6 DataObjectInstance Struct Reference

Struct to describe an Object instance with its sequence number within a topic or object type.

```
#include <protocol.hpp>
```

Public Member Functions

- `DataObjectInstance` ()

Public Attributes

- enum `Topic object_type`
- `uint16_t object_sequence_nr`

6.6.1 Detailed Description

Struct to describe an Object instance with its sequence number within a topic or object type.

Todo o Umbenennen, eher `DataObjectIdentifier`

6.6.2 Constructor & Destructor Documentation

6.6.2.1 DataObjectInstance()

```
DataObjectInstance::DataObjectInstance ( ) [inline]
```

6.6.3 Member Data Documentation

6.6.3.1 object_sequence_nr

```
uint16_t DataObjectInstance::object_sequence_nr
```

6.6.3.2 object_type

```
enum Topic DataObjectInstance::object_type
```

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

- include/adom/protocol.hpp

6.7 DataRequest Class Reference

[DataRequest](#) is used by a remote (subscribed) directory (`reader_application_`) to request specific blocks of a sample object managed by the data's (publihser/) home directory (`home_directory_`) via a `block_validity_matrix`.

```
#include <protocol.hpp>
```

Public Member Functions

- [DataRequest](#) ()
- [DataRequest](#) ([EntityDescriptor](#) reader_application, [EntityDescriptor](#) home_directory, [DataObjectInstance](#) object_instance, uint16_t object_block_count, int *object_block_validity, int request_id)
- void [dataRequestToNet](#) (char *msg)

This function is called to serialize [DataRequest](#) for transport.
- void [netToDataRequest](#) (char *msg)

This function is called to deserialize received [DataRequest](#).
- void [clear](#) ()

This function is called to clear all attributes of the [DataRequest](#) for reuse.
- void [print](#) ()

This function prints out the [DataRequest](#).

Public Attributes

- uint16_t [request_id_](#)
- EntityDescriptor [reader_application_](#)
- EntityDescriptor [home_directory_](#)
- DataObjectInstance [object_instance_](#)
- uint16_t [object_block_count_](#)
- std::vector< uint8_t > [object_block_validity_](#)
- uint16_t [message_length_](#)

6.7.1 Detailed Description

[DataRequest](#) is used by a remote (subscribed) directory ([reader_application_](#)) to request specific blocks of a sample object managed by the data's (publiher/) home directory ([home_directory_](#)) via a [block_validity_matrix](#).

6.7.2 Constructor & Destructor Documentation

6.7.2.1 DataRequest() [1/2]

```
DataRequest::DataRequest ( ) [inline]
```

6.7.2.2 DataRequest() [2/2]

```
DataRequest::DataRequest (
    EntityDescriptor reader_application,
    EntityDescriptor home_directory,
    DataObjectInstance object_instance,
    uint16_t object_block_count,
    int * object_block_validity,
    int request_id ) [inline]
```

6.7.3 Member Function Documentation

6.7.3.1 clear()

```
void DataRequest::clear ( )
```

This function is called to clear all attributes of the [DataRequest](#) for reuse.

6.7.3.2 dataRequestToNet()

```
void DataRequest::dataRequestToNet (
    char * msg )
```

This function is called to serialize [DataRequest](#) for transport.

Parameters

<i>msg</i>	Pointer to buffer to write the serialized request into
------------	--

6.7.3.3 netToDataRequest()

```
void DataRequest::netToDataRequest (
    char * msg )
```

This function is called to deserialize received [DataRequest](#).

Parameters

<i>msg</i>	Pointer to buffer to the serialized request
------------	---

6.7.3.4 print()

```
void DataRequest::print ( )
```

This function prints out the [DataRequest](#).

6.7.4 Member Data Documentation

6.7.4.1 home_directory_

[EntityDescriptor](#) DataRequest::home_directory_

6.7.4.2 message_length_

uint16_t DataRequest::message_length_

6.7.4.3 object_block_count_

uint16_t DataRequest::object_block_count_

6.7.4.4 object_block_validity_

```
std::vector<uint8_t> DataRequest::object_block_validity_
```

6.7.4.5 object_instance_

```
DataObjectInstance DataRequest::object_instance_
```

6.7.4.6 reader_application_

```
EntityDescriptor DataRequest::reader_application_
```

6.7.4.7 request_id_

```
uint16_t DataRequest::request_id_
```

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

- include/adom/protocol.hpp
- src/cpp/adom/protocol.cpp

6.8 DataTransport Class Reference

[DataTransport](#) is used by a home directory (publisher) to transport requested blocks of a sample object to a remote (subscribed) directory.

```
#include <protocol.hpp>
```

Public Member Functions

- [DataTransport](#) ()
- [DataTransport](#) ([DataBlockHeader](#) block_header, unsigned char *data, uint16_t associated_request_id)
- void [dataTransportToNet](#) (char *msg)

This function is called to serialize [DataTransport](#) for transport.
- void [netToDataTransport](#) (char *msg)

This function is called to deserialize received [DataTransport](#).
- void [clear](#) ()

This function is called to clear all attributes of the [DataTransport](#) for reuse.
- void [print](#) ()

This function prints out the [DataTransport](#).

Public Attributes

- uint16_t [associated_request_id_](#)
- [DataObjectInstance](#) [associated_object_](#)
- uint16_t [block_id_](#)
- uint16_t [block_size_](#)
- std::vector< char > [data_](#)
- uint16_t [message_length_](#)

6.8.1 Detailed Description

[DataTransport](#) is used by a home directory (publisher) to transport requested blocks of a sample object to a remote (subscribed) directory.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 DataTransport() [1/2]

```
DataTransport::DataTransport ( ) [inline]
```

6.8.2.2 DataTransport() [2/2]

```
DataTransport::DataTransport (
    DataBlockHeader block_header,
    unsigned char * data,
    uint16_t associated_request_id ) [inline]
```

6.8.3 Member Function Documentation

6.8.3.1 clear()

```
void DataTransport::clear ( )
```

This function is called to clear all attributes of the [DataTransport](#) for reuse.

6.8.3.2 dataTransportToNet()

```
void DataTransport::dataTransportToNet (
    char * msg )
```

This function is called to serialize [DataTransport](#) for transport.

Parameters

<i>msg</i>	Pointer to buffer to write the serialized DataTransport into
------------	--

6.8.3.3 netToDataTransport()

```
void DataTransport::netToDataTransport (
    char * msg )
```

This function is called to deserialize received [DataTransport](#).

Parameters

<i>msg</i>	Pointer to buffer to the serialized DataTransport
------------	---

6.8.3.4 print()

```
void DataTransport::print ( )
```

This function prints out the [DataTransport](#).

6.8.4 Member Data Documentation**6.8.4.1 associated_object_**

[DataObjectInstance](#) DataTransport::associated_object_

6.8.4.2 associated_request_id_

uint16_t DataTransport::associated_request_id_

6.8.4.3 block_id_

uint16_t DataTransport::block_id_

6.8.4.4 block_size_

```
uint16_t DataTransport::block_size_
```

6.8.4.5 data_

```
std::vector<char> DataTransport::data_
```

6.8.4.6 message_length_

```
uint16_t DataTransport::message_length_
```

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

- include/adom/protocol.hpp
- src/cpp/adom/protocol.cpp

6.9 Directory Class Reference

```
#include <directory.hpp>
```

Public Member Functions

- **Directory** ([EntityDescriptor](#) entity_descriptor, [SafeQueue](#)< std::pair< [DataRequest](#), udp::endpoint >> &directory_request_queue, [SafeQueue](#)< [DataTransport](#) > &directory_reply_queue, [SafeQueue](#)< std::pair< [DataAnnouncement](#), udp::endpoint >> &directory_announcement_queue)
Construct a new [Directory](#) object.
- void **initiate** ()
Initialize the new directory. Handler threads are started and an ingress endpoint for communication is created.
- void **addNewObjectSample** (std::shared_ptr< [ObjectSample](#) > new_sample)
This function is called at the application interface, as soon as new data is available. New data is either added by an user application directly, or through the reception of a data announcement.
- std::vector< uint16_t > **checkAvailability** (std::shared_ptr< [ObjectSample](#) > sample, std::vector< uint16_t > block_ids)
This function is called to check whether a given set of blocks (identified by their IDs) for a given object sample is currently locally available.
- int **sendDataRequest** ([EntityDescriptor](#) home_directory, std::shared_ptr< [ObjectSample](#) >, std::vector< uint16_t > block_ids)
This function is called from within a read process triggered by the application. After the data to be read and the associated blocks are determined, this function is called to initiate the process of sending the required data requests.
- std::shared_ptr< [ObjectSample](#) > **getLastObjectSample** ([Topic](#) type)
This function is called to obtain a shared pointer to the most up-to-date sample for a specified topic that is managed by the directory.
- std::shared_ptr< [ObjectSample](#) > **getSample** ([Topic](#) type, int sequence_nr)

This function is called to obtain a shared pointer to the sample for a specified topic and a specified sequence number that is managed by the directory.

- void [printTrackedObjectSamples](#) ()
Print out all objects tracked and cached by the directory.
- int [getNumberOfTrackedObjects](#) (Topic topic)
This function returns the number of tracked samples for a given topic.
- uint16_t [getLatestSequencenumberOfTrackedTopic](#) (Topic topic)
This funtion returns the sequence number of the most up-to-date sample of a specified topic.
- void [setRequestedBlocks](#) (int request_id, const std::set< uint16_t > &requested_blocks)
Set the Expected Blocks object for a specific request.
- void [join](#) ()
Joining all threads for smooth exit.
- void [terminate](#) ()
Terminating all threads.
- void [freeAllObjects](#) ()
Freeing all Objects.
- void [stop](#) ()
Stopping all processes within directory.

Public Attributes

- [EntityDescriptor other_directory_descriptor_](#)
information/ descriptor on other directories. Temporary solution as compensatoin for a subscriber list, that is currently missing as no discovery process is implemented
- std::mutex [request_tracking_lock_](#)
- std::unordered_map< int, [RequestTracking](#) > [request_tracking_](#)
List to track received Blocks for pending requests.

6.9.1 Constructor & Destructor Documentation

6.9.1.1 Directory()

```
Directory::Directory (
    EntityDescriptor entity_descriptor,
    SafeQueue< std::pair< DataRequest, udp::endpoint >> & directory_request_queue,
    SafeQueue< DataTransport > & directory_reply_queue,
    SafeQueue< std::pair< DataAnnouncement, udp::endpoint >> & directory_announcement↵
_queue ) [inline]
```

Construct a new [Directory](#) object.

Parameters

<code>entity_descriptor</code>	
--------------------------------	--

6.9.2 Member Function Documentation

6.9.2.1 addNewObjectSample()

```
void Directory::addNewObjectSample (
    std::shared_ptr< ObjectSample > new_sample )
```

This function is called at the application interface, as soon as new data is available. New data is either added by an user application directly, or through the reception of a data announcement.

This function specifically is called, when the sample sequence number is not directly given, e.g. by the user application. Therefore, the last tracked sequence number has to be determined.

Parameters

<i>new_sample</i>	shared Pointer to new ObjectSample to be tracked and distributed if requested
-------------------	---

6.9.2.2 checkAvailability()

```
std::vector< uint16_t > Directory::checkAvailability (
    std::shared_ptr< ObjectSample > sample,
    std::vector< uint16_t > block_ids )
```

This function is called to check whether a given set of blocks (identified by their IDs) for a given object sample is currently locally available.

Parameters

<i>sample</i>	associated object sample
<i>block_ids</i>	given set of requested block ID

6.9.2.3 freeAllObjects()

```
void Directory::freeAllObjects ( )
```

Freeing all Objects.

6.9.2.4 getLastObjectSample()

```
std::shared_ptr< ObjectSample > Directory::getLastObjectSample (
    Topic type )
```

This function is called to obtain a shared pointer to the most up-to-date sample for a specified topic that is managed by the directory.

Parameters

<i>type</i>	specified object sample topic
-------------	-------------------------------

Returns

std::shared_ptr<ObjectSample> shared pointer to most up-to-date sample

6.9.2.5 getLatestSequencenumberOfTrackedTopic()

```
uint16_t Directory::getLatestSequencenumberOfTrackedTopic (
    Topic type )
```

This funtion returns the sequece number of the most up-to-date sample of a specified topic.

Parameters

<i>type</i>	specified object sample topic
-------------	-------------------------------

Returns

uint16_t latest sequence number

6.9.2.6 getNumberOfTrackedObjects()

```
int Directory::getNumberOfTrackedObjects (
    Topic topic )
```

This function returns the number of tracked samples for a given topic.

Parameters

<i>topic</i>	Topic to check the tracked sample number on
--------------	---

Returns

int Number of tracked samples for the given topic

6.9.2.7 getSample()

```
std::shared_ptr< ObjectSample > Directory::getSample (
    Topic type,
    int sequence_nr )
```

This function is called to obtain a shared pointer to the sample for a specified topic and a specified sequence number that is managed by the directory.

Parameters

<i>type</i>	specified object sample topic
<i>sequence_↔ _nr</i>	specified sequence number

Returns

std::shared_ptr<ObjectSample> shared pointer to most up-to-date sample

6.9.2.8 initiate()

```
void Directory::initiate ( )
```

Initialize the new directory. Handler threads are started and an ingress endpoint for communication is created.

6.9.2.9 join()

```
void Directory::join ( )
```

Joining all threads for smooth exit.

6.9.2.10 printTrackedObjectSamples()

```
void Directory::printTrackedObjectSamples ( )
```

Print out all objects tracked and cached by the directory.

6.9.2.11 sendDataRequest()

```
int Directory::sendDataRequest (
    EntityDescriptor home_directory,
    std::shared_ptr< ObjectSample > sample,
    std::vector< uint16_t > block_ids )
```

This function is called from within a read process triggered by the application. After the data to be read and the associated blocks are determined, this function is called to initiate the process of sending the required data requests.

Todo make private

Parameters

<i>home_directory</i>	Home directory for the data that is requested
<i>sample</i>	Affiliated sample the read process is directed to

Returns

Request ID

Todo make private

Parameters

<i>home_directory</i>	Home directory for the data that is requested
<i>sample</i>	Affiliated sample the read process is directed to

Returns

Request ID

6.9.2.12 setRequestedBlocks()

```
void Directory::setRequestedBlocks (
    int request_id,
    const std::set< uint16_t > & requested_blocks )
```

Set the Expected Blocks object for a specific request.

Parameters

<i>request_id</i>	associated request ID
<i>expected_blocks</i>	given requested and therefore expected blocks

6.9.2.13 stop()

```
void Directory::stop ( )
```

Stopping all processes within directory.

6.9.2.14 terminate()

```
void Directory::terminate ( )
```

Terminating all threads.

Joining all threads for smooth exit.

6.9.3 Member Data Documentation

6.9.3.1 other_directory_descriptor_

```
EntityDescriptor Directory::other_directory_descriptor_
```

information/ descriptor on other directories. Temporary solution as compensatoin for a subscriber list, that is currently missing as no discovery process is implemented

Todo make private

6.9.3.2 request_tracking_

```
std::unordered_map<int, RequestTracking> Directory::request_tracking_
```

List to track received Blocks for pending requests.

6.9.3.3 request_tracking_lock_

```
std::mutex Directory::request_tracking_lock_
```

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

- include/adom/directory.hpp
- src/cpp/adom/directory.cpp

6.10 EntityDescriptor Struct Reference

This struct is utilized to describe an Entity of a directory. This struct must be filled prior to setting up the directory struct itself.

```
#include <protocol.hpp>
```

Public Member Functions

- [EntityDescriptor](#) ()

Public Attributes

- uint16_t [entity_id](#)
- char [ip_address](#) [[string_length](#)]
- uint16_t [ingress_port](#)

6.10.1 Detailed Description

This struct is utilized to describe an Entity of a directory. This struct must be filled prior to setting up the directory struct itself.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 EntityDescriptor()

```
EntityDescriptor::EntityDescriptor ( ) [inline]
```

6.10.3 Member Data Documentation

6.10.3.1 entity_id

```
uint16_t EntityDescriptor::entity_id
```

6.10.3.2 ingress_port

```
uint16_t EntityDescriptor::ingress_port
```

6.10.3.3 ip_address

```
char EntityDescriptor::ip_address[string\_length]
```

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

- [include/adom/protocol.hpp](#)

6.11 ObjectSample Class Reference

Class to describe an [ObjectSample](#).

```
#include <directory.hpp>
```

Public Member Functions

- [ObjectSample](#) ([EntityDescriptor](#) home_directory, [Topic](#) topic, uint16_t updated_sequence_nr, unsigned char *data, [Structure](#) structure)
- [EntityDescriptor](#) getHomeDirectory ()
- [Topic](#) getTopic ()
- uint16_t getSequenceNumber ()
- uint16_t getBlockCount ()
- [Structure](#) getStructure ()
- [DataBlockHeader](#) getHeader (int block_id)
- std::vector< [DataBlockHeader](#) > getHeader ()
- int isAvailable (int block_id)
- void makeAvailable (int block_id)
- void printObjectSample ()

Public Attributes

- std::vector< unsigned char > [object_sample_data_](#)

6.11.1 Detailed Description

Class to describe an [ObjectSample](#).

6.11.2 Constructor & Destructor Documentation

6.11.2.1 ObjectSample()

```
ObjectSample::ObjectSample (
    EntityDescriptor home_directory,
    Topic topic,
    uint16_t updated_sequence_nr,
    unsigned char * data,
    Structure structure ) [inline]
```

6.11.3 Member Function Documentation

6.11.3.1 getBlockCount()

```
uint16_t ObjectSample::getBlockCount ( ) [inline]
```

6.11.3.2 getHeader() [1/2]

```
std::vector<DataBlockHeader> ObjectSample::getHeader ( ) [inline]
```

6.11.3.3 getHeader() [2/2]

```
DataBlockHeader ObjectSample::getHeader (
    int block_id ) [inline]
```

6.11.3.4 getHomeDirectory()

```
EntityDescriptor ObjectSample::getHomeDirectory ( ) [inline]
```

6.11.3.5 getSequenceNumber()

```
uint16_t ObjectSample::getSequenceNumber ( ) [inline]
```

6.11.3.6 getStructure()

```
Structure ObjectSample::getStructure ( ) [inline]
```

6.11.3.7 getTopic()

```
Topic ObjectSample::getTopic ( ) [inline]
```

6.11.3.8 `isAvailable()`

```
int ObjectSample::isAvailable (
    int block_id ) [inline]
```

6.11.3.9 `makeAvailable()`

```
void ObjectSample::makeAvailable (
    int block_id ) [inline]
```

6.11.3.10 `printObjectSample()`

```
void ObjectSample::printObjectSample ( ) [inline]
```

6.11.4 Member Data Documentation

6.11.4.1 `object_sample_data_`

```
std::vector<unsigned char> ObjectSample::object_sample_data_
```

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

- [include/adom/directory.hpp](#)

6.12 RequestTracking Struct Reference

```
#include <directory.hpp>
```

Public Attributes

- `std::set< uint16_t >` [received_blocks](#)
- `std::set< uint16_t >` [requested_blocks](#)
- `std::condition_variable` [cv](#)

6.12.1 Member Data Documentation

6.12.1.1 cv

```
std::condition_variable RequestTracking::cv
```

6.12.1.2 received_blocks

```
std::set<uint16_t> RequestTracking::received_blocks
```

6.12.1.3 requested_blocks

```
std::set<uint16_t> RequestTracking::requested_blocks
```

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

- include/adom/[directory.hpp](#)

6.13 SafeQueue< T > Class Template Reference

A thread safe queue.

```
#include <safe_queue.hpp>
```

Public Member Functions

- [SafeQueue](#) (void)
Construct a new (empty) Safe Queue object.
- [~SafeQueue](#) (void)
Destroy the Safe Queue object (default)
- void [enqueue](#) (T value)
Thread safe wrapper for std::queue push. A thread waiting on dequeue is notified after the enqueue operation.
- T [dequeue](#) (void)
Get the "front"-element. If the queue is empty, wait till an element is available.
- T [dequeue](#) (bool blocking=true)
Get the "front"-element. (!) Note: A non blocking call to an empty queue will lead to undefined behavior. Thus the [empty\(\)](#) method needs to be called first. (!) Note: A clean implementation would use exceptions to avoid a call to an empty queue.
- T [dequeue](#) (bool blocking=true, std::chrono::milliseconds timeout=std::chrono::milliseconds::zero())
- bool [empty](#) ()
Thread safe wrapper for std::queue [empty\(\)](#)

Protected Attributes

- `std::queue< T >` [safe_queue](#)
The underlying queue which is accessed via the `queue_lock` (mutex)
- `std::mutex` [queue_lock](#)
The mutex controlling the concurrent queue access.
- `std::condition_variable` [queue_event](#)
Condition variable used for an enqueue event. The blocking dequeue call waits for an enqueue event if the queue is empty.

6.13.1 Detailed Description

```
template<class T>
class SafeQueue< T >
```

A thread safe queue.

Template Parameters

<code>T</code>	the data type to be stored in the queue
----------------	---

6.13.2 Constructor & Destructor Documentation

6.13.2.1 SafeQueue()

```
template<class T >
SafeQueue< T >::SafeQueue (
    void ) [inline]
```

Construct a new (empty) Safe Queue object.

6.13.2.2 ~SafeQueue()

```
template<class T >
SafeQueue< T >::~~SafeQueue (
    void ) [inline]
```

Destroy the Safe Queue object (default)

6.13.3 Member Function Documentation

6.13.3.1 dequeue() [1/3]

```
template<class T >
T SafeQueue< T >::dequeue (
    bool blocking = true ) [inline]
```

Get the "front"-element. (!) Note: A non blocking call to an empty queue will lead to undefined behavior. Thus the `empty()` method needs to be called first. (!) Note: A clean implementation would use exceptions to avoid a call to an empty queue.

Parameters

<i>blocking</i>	
-----------------	--

Returns

T The dequeued value

6.13.3.2 dequeue() [2/3]

```
template<class T >
T SafeQueue< T >::dequeue (
    bool blocking = true,
    std::chrono::milliseconds timeout = std::chrono::milliseconds::zero() ) [inline]
```

6.13.3.3 dequeue() [3/3]

```
template<class T >
T SafeQueue< T >::dequeue (
    void ) [inline]
```

Get the "front"-element. If the queue is empty, wait till an element is available.

Returns

T The dequeued value

6.13.3.4 empty()

```
template<class T >
bool SafeQueue< T >::empty ( ) [inline]
```

Thread safe wrapper for `std::queue` `empty()`

Returns

true empty queue

false non empty queue

6.13.3.5 enqueue()

```
template<class T >
void SafeQueue< T >::enqueue (
    T value ) [inline]
```

Thread safe wrapper for std::queue push. A thread waiting on dequeue is notified after the enqueue operation.

Parameters

<i>value</i>	the value to be enqueued
--------------	--------------------------

6.13.4 Member Data Documentation

6.13.4.1 queue_event

```
template<class T >
std::condition_variable SafeQueue< T >::queue_event [protected]
```

Condition variable used for an enqueue event. The blocking dequeue call waits for an enqueue event if the queue is empty.

6.13.4.2 queue_lock

```
template<class T >
std::mutex SafeQueue< T >::queue_lock [mutable], [protected]
```

The mutex controlling the concurrent queue access.

6.13.4.3 safe_queue

```
template<class T >
std::queue<T> SafeQueue< T >::safe_queue [protected]
```

The underlying queue which is accessed via the queue_lock (mutex)

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

- include/adom/[safe_queue.hpp](#)

6.14 socket_endpoint Struct Reference

Describes the parameters of an boost asio endpoint. Used for passing endpoints to functions.

```
#include <socket_endpoint.hpp>
```

Public Member Functions

- [socket_endpoint](#) ()
- [socket_endpoint](#) (std::string ip, int p)

Public Attributes

- std::string [ip_addr](#)
- int [port](#)

6.14.1 Detailed Description

Describes the parameters of an boost asio endpoint. Used for passing endpoints to functions.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 socket_endpoint() [1/2]

```
socket_endpoint::socket_endpoint ( ) [inline]
```

6.14.2.2 socket_endpoint() [2/2]

```
socket_endpoint::socket_endpoint (
    std::string ip,
    int p ) [inline]
```

6.14.3 Member Data Documentation

6.14.3.1 ip_addr

```
std::string socket_endpoint::ip_addr
```

6.14.3.2 port

```
int socket_endpoint::port
```

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

- include/adom/[socket_endpoint.hpp](#)

6.15 Structure Struct Reference

To decompose the object sample into data blocks, that are beneficial to the data type or user application's use of the data, the user should specify the proper structure of the decomposed object sample.

```
#include <translation.h>
```

Public Member Functions

- [Structure](#) ()
- [Structure](#) ([StructureType](#) type, uint16_t [block_rows](#), uint16_t [block_cols](#), uint16_t [object_height](#), uint16_t [object_width](#), uint16_t [object_channels](#))

Public Attributes

- [StructureType](#) type
- uint16_t [block_rows](#)
- uint16_t [block_cols](#)
- uint16_t [object_height](#)
- uint16_t [object_width](#)
- uint16_t [object_channels](#)

6.15.1 Detailed Description

To decompose the object sample into data blocks, that are beneficial to the data type or user application's use of the data, the user should specify the proper structure of the decomposed object sample.

1D data structure: Data blocks are composed of one (or several successive) memory "lines" (equals standard hardware data caching) –For 1D data, [block_line_width](#) equals about 1kB to fill an ethernet frame and [block_line_count](#) is 1.– For 1D data, [object_height](#) and [block_rows](#) is 1.

2D data structure: Data blocks are composed of several NON successive memory "lines" –For 2D data, the following applies: [block_line_width](#) * [block_line_count](#) <= 1kB and [block_line_count](#) > 1.–

6.15.2 Constructor & Destructor Documentation

6.15.2.1 Structure() [1/2]

```
Structure::Structure ( ) [inline]
```

6.15.2.2 Structure() [2/2]

```
Structure::Structure (
    StructureType type,
    uint16_t block_rows,
    uint16_t block_cols,
    uint16_t object_height,
    uint16_t object_width,
    uint16_t object_channels ) [inline]
```

6.15.3 Member Data Documentation

6.15.3.1 block_cols

```
uint16_t Structure::block_cols
```

6.15.3.2 block_rows

```
uint16_t Structure::block_rows
```

6.15.3.3 object_channels

```
uint16_t Structure::object_channels
```

6.15.3.4 object_height

```
uint16_t Structure::object_height
```

6.15.3.5 object_width

```
uint16_t Structure::object_width
```

6.15.3.6 type

```
StructureType Structure::type
```

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

- include/adom/[translation.h](#)

Chapter 7

File Documentation

7.1 include/adom/application_interface.hpp File Reference

```
#include <unistd.h>
#include <string.h>
#include <string>
#include <vector>
#include "protocol.hpp"
#include "directory.hpp"
#include "log.hpp"
```

Classes

- class [ApplicationInterface](#)

7.2 include/adom/directory.hpp File Reference

```
#include <unistd.h>
#include <string.h>
#include <string>
#include <condition_variable>
#include <unordered_map>
#include <mutex>
#include <vector>
#include <set>
#include <list>
#include <thread>
#include <boost/asio.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <adom/safe_queue.hpp>
#include <adom/protocol.hpp>
#include <adom/parameters.h>
#include <adom/log.hpp>
```

Classes

- struct [RequestTracking](#)
- class [ObjectSample](#)
 - Class to describe an [ObjectSample](#).
- class [Directory](#)

7.3 include/adom/log.hpp File Reference

```
#include <boost/log/core.hpp>
#include <boost/log/trivial.hpp>
#include <boost/log/expressions.hpp>
#include <boost/log/sinks/text_file_backend.hpp>
#include <boost/log/utility/setup/file.hpp>
#include <boost/log/utility/setup/console.hpp>
#include <boost/log/utility/setup/common_attributes.hpp>
#include <boost/log/sources/severity_logger.hpp>
#include <boost/log/sources/record_ostream.hpp>
#include <boost/log/utility/setup/settings.hpp>
#include <boost/log/attributes/timer.hpp>
#include <boost/log/attributes/named_scope.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/log/attributes/clock.hpp>
#include <boost/log/support/date_time.hpp>
#include <adom/parameters.h>
#include <cstdlib>
#include <iostream>
```

Namespaces

- [adom](#)

Macros

- #define [logTrace](#)(msg)
 - Preprocessor wrapper for boost logging library.
- #define [logDebug](#)(msg)
- #define [logInfo](#)(msg)
- #define [logWarning](#)(msg)
- #define [logError](#)(msg)
- #define [logFatal](#)(msg)
- #define [AppLog](#)(msg)

Functions

- void [adom::init_file_log](#) (std::string log_prefix, std::string log_suffix)
- void [adom::init_console_log](#) ()

7.3.1 Macro Definition Documentation

7.3.1.1 AppLog

```
#define AppLog(  
    msg )
```

7.3.1.2 logDebug

```
#define logDebug(  
    msg )
```

7.3.1.3 logError

```
#define logError(  
    msg )
```

7.3.1.4 logFatal

```
#define logFatal(  
    msg )
```

7.3.1.5 logInfo

```
#define logInfo(  
    msg )
```

7.3.1.6 logTrace

```
#define logTrace(  
    msg )
```

Preprocessor wrapper for boost logging library.

7.3.1.7 logWarning

```
#define logWarning(  
    msg )
```

7.4 include/adom/opencv_helper.h File Reference

```
#include <unistd.h>  
#include <string.h>  
#include <string>  
#include <vector>  
#include <adom/translation.h>
```

Functions

- `uint16_t findBlockFromAddress` (const unsigned char *readDataAddress, const unsigned char *objectStartData, [Structure](#) data_structure)
Finds associated data block for a given data address and returns the number of the associated block.
- `uint16_t findBlock` (int x, int y, [Structure](#) data_structure)
Finds associated data block for a given data address and returns the number of the associated block.
- `std::vector< uint16_t > findBlocksForReadAccess` (int first_pixel_x, int first_pixel_y, int last_pixel_x, int last_pixel_y, [Structure](#) data_structure)
Finds associated data blocks for a read access to an image that is specified by the first pixel and the last pixel.

7.4.1 Function Documentation

7.4.1.1 findBlock()

```
uint16_t findBlock (  
    int x,  
    int y,  
    Structure data_structure ) [inline]
```

Finds associated data block for a given data address and returns the number of the associated block.

Parameters

<i>readDataAddress</i>	given data address (through read process)
<i>objectStartData</i>	start address of data objects
<i>data_structure</i>	internal structure of the data object and how it is cached

Returns

int

7.4.1.2 findBlockFromAddress()

```
uint16_t findBlockFromAddress (
    const unsigned char * readDataAddress,
    const unsigned char * objectStartData,
    Structure data_structure ) [inline]
```

Finds associated data block for a given data address and returns the number of the associated block.

Parameters

<i>readDataAddress</i>	given data address (through read process)
<i>objectStartData</i>	start address of data objects
<i>data_structure</i>	internal structure of the data object and how it is cached

Returns

int

7.4.1.3 findBlocksForReadAccess()

```
std::vector<uint16_t> findBlocksForReadAccess (
    int first_pixel_x,
    int first_pixel_y,
    int last_pixel_x,
    int last_pixel_y,
    Structure data_structure ) [inline]
```

Finds associated data blocks for a read access to an image that is specified by the first pixel and the last pixel.

Parameters

<i>first_pixel_x</i>	x coordinate of start pixel
<i>first_pixel_y</i>	y coordinate of start pixel
<i>last_pixel_x</i>	x coordinate of end pixel
<i>last_pixel_y</i>	y coordinate of end pixel
<i>data_structure</i>	structure of the image, e.g. number of columns and rows...

Returns

std::vector<uint16_t> block IDs of the associated blocks

7.5 include/adom/parameters.h File Reference

```
#include <unistd.h>
#include <string.h>
#include <string>
#include <boost/asio.hpp>
```

Macros

- #define [PROTOCOL_TEST](#) 1
- #define [MAX_COUNT_MANAGED_OBJECTS_PER_TOPIC](#) 3
- #define [PRINTF_BINARY_PATTERN_INT8](#) "%C%C%C%C%C%C%C%C"
- #define [PRINTF_BYTE_TO_BINARY_INT8](#)(i)

Enumerations

- enum [Topic](#) { [NONE](#) = 0 , [IMAGE](#) = 1 }

Enum to describe different topics.

Variables

- const std::string [log_directory](#) = "/ApplicationDataObjectManagement/examples/adom_logs/"
- const char [publisher_ip](#) [20] = "192.168.3.100"
- const char [subscriber_ip](#) [20] = "192.168.3.101"
- const uint16_t [home_dir_ingress_port](#) = 5400
- const uint16_t [req_dir_ingress_port](#) = 5402
- const uint16_t [string_length](#) = 20
- const uint16_t [adom_message_overhead](#) = 8
- const uint16_t [max_buffer_length](#) = 1400
- const uint16_t [TOTAL_BLOCK_SIZE](#) = 1200
- const uint16_t [HD_BLOCKS_IN_ROW](#) = 64
- const uint16_t [HD_BLOCK_ROWS](#) = 36
- const uint16_t [HD_V_PIXEL_PER_IMAGE](#) = 720
- const uint16_t [HD_H_PIXEL_PER_IMAGE](#) = 1280
- const uint16_t [FULL_HD_BLOCKS_IN_ROW](#) = 96
- const uint16_t [FULL_HD_BLOCK_ROWS](#) = 54
- const uint16_t [FULL_HD_V_PIXEL_PER_IMAGE](#) = 1080
- const uint16_t [FULL_HD_H_PIXEL_PER_IMAGE](#) = 1920
- const uint16_t [V_PIXEL_PER_BLOCK](#) = 20
- const uint16_t [H_PIXEL_PER_BLOCK](#) = 20
- const uint16_t [CHANNEL_NUMBER](#) = 3
- const uint16_t [MAX_NUM_TRACKED_SAMPLES](#) = 5
- const uint16_t [EMPTY_QUEUE_ERROR](#) = 20

7.5.1 Macro Definition Documentation

7.5.1.1 MAX_COUNT_MANAGED_OBJECTS_PER_TOPIC

```
#define MAX_COUNT_MANAGED_OBJECTS_PER_TOPIC 3
```

7.5.1.2 PRINTF_BINARY_PATTERN_INT8

```
#define PRINTF_BINARY_PATTERN_INT8 "%c%c%c%c%c%c%c%c"
```

7.5.1.3 PRINTF_BYTE_TO_BINARY_INT8

```
#define PRINTF_BYTE_TO_BINARY_INT8(  
    i )
```

Value:

```
((i) & 0x8011) ? '1' : '0', \  
((i) & 0x4011) ? '1' : '0', \  
((i) & 0x2011) ? '1' : '0', \  
((i) & 0x1011) ? '1' : '0', \  
((i) & 0x0811) ? '1' : '0', \  
((i) & 0x0411) ? '1' : '0', \  
((i) & 0x0211) ? '1' : '0', \  
((i) & 0x0111) ? '1' : '0'
```

7.5.1.4 PROTOCOL_TEST

```
#define PROTOCOL_TEST 1
```

7.5.2 Enumeration Type Documentation

7.5.2.1 Topic

```
enum Topic
```

Enum to describe different topics.

Enumerator

NONE	
IMAGE	

7.5.3 Variable Documentation

7.5.3.1 adom_message_overhead

```
const uint16_t adom_message_overhead = 8
```

7.5.3.2 CHANNEL_NUMBER

```
const uint16_t CHANNEL_NUMBER = 3
```

7.5.3.3 EMPTY_QUEUE_ERROR

```
const uint16_t EMPTY_QUEUE_ERROR = 20
```

7.5.3.4 FULL_HD_BLOCK_ROWS

```
const uint16_t FULL_HD_BLOCK_ROWS = 54
```

7.5.3.5 FULL_HD_BLOCKS_IN_ROW

```
const uint16_t FULL_HD_BLOCKS_IN_ROW = 96
```

7.5.3.6 FULL_HD_H_PIXEL_PER_IMAGE

```
const uint16_t FULL_HD_H_PIXEL_PER_IMAGE = 1920
```

7.5.3.7 FULL_HD_V_PIXEL_PER_IMAGE

```
const uint16_t FULL_HD_V_PIXEL_PER_IMAGE = 1080
```

7.5.3.8 H_PIXEL_PER_BLOCK

```
const uint16_t H_PIXEL_PER_BLOCK = 20
```

7.5.3.9 HD_BLOCK_ROWS

```
const uint16_t HD_BLOCK_ROWS = 36
```

7.5.3.10 HD_BLOCKS_IN_ROW

```
const uint16_t HD_BLOCKS_IN_ROW = 64
```

7.5.3.11 HD_H_PIXEL_PER_IMAGE

```
const uint16_t HD_H_PIXEL_PER_IMAGE = 1280
```

7.5.3.12 HD_V_PIXEL_PER_IMAGE

```
const uint16_t HD_V_PIXEL_PER_IMAGE = 720
```

7.5.3.13 home_dir_ingress_port

```
const uint16_t home_dir_ingress_port = 5400
```

7.5.3.14 log_directory

```
const std::string log_directory = "/ApplicationDataObjectManagement/examples/adom_logs/"
```

7.5.3.15 max_buffer_length

```
const uint16_t max_buffer_length = 1400
```

7.5.3.16 MAX_NUM_TRACKED_SAMPLES

```
const uint16_t MAX_NUM_TRACKED_SAMPLES = 5
```

7.5.3.17 publisher_ip

```
const char publisher_ip[20] = "192.168.3.100"
```

7.5.3.18 req_dir_ingress_port

```
const uint16_t req_dir_ingress_port = 5402
```

7.5.3.19 string_length

```
const uint16_t string_length = 20
```

7.5.3.20 subscriber_ip

```
const char subscriber_ip[20] = "192.168.3.101"
```

7.5.3.21 TOTAL_BLOCK_SIZE

```
const uint16_t TOTAL_BLOCK_SIZE = 1200
```

7.5.3.22 V_PIXEL_PER_BLOCK

```
const uint16_t V_PIXEL_PER_BLOCK = 20
```

7.6 include/adom/protocol.hpp File Reference

```
#include <unistd.h>
#include <string.h>
#include <string>
#include <iostream>
#include <vector>
#include <adom/parameters.h>
#include <adom/translation.h>
#include <adom/debug.h>
```

Classes

- struct [EntityDescriptor](#)
This struct is utilized to describe an Entity of a directory. This struct must be filled prior to setting up the directory struct itself.
- struct [DataObjectInstance](#)
Struct to describe an Object instance with its sequence number within a topic or object type.
- struct [DataBlockHeader](#)
Struct to describe a Data Block, that is part of an Object instance.
- class [AdomMessage](#)
Application Data Object Management Messages are used as Wrapper Data Announcements, for Data Request, and Date Transports.
- class [DataRequest](#)
[DataRequest](#) is used by a remote (subscribed) directory ([reader_application_](#)) to request specific blocks of a sample object managed by the data's (publihser/) home directory ([home_directory_](#)) via a [block_validity_matrix](#).
- class [DataTransport](#)
[DataTransport](#) is used by a home directory (publisher) to transport requested blocks of a sample object to a remote (subscribed) directory.
- class [DataAnnouncement](#)
[DataAnnouncement](#) is used by a home directory (publisher) to inform a remote (subscribed) directory of a new available sample.
- struct [AppRequest](#)

Enumerations

- enum [MessageType](#) { [UNKNOWN](#) = 0 , [REQUEST](#) = 1 , [REPLY](#) = 2 , [ANNOUNCEMENT](#) }
 - enum [BlockValidity](#) { [INVALID](#) = 0 , [REQUESTED](#) , [VALID](#) , [SHARED](#) }
- Enum to describe the status of a block within a validity matrix of an object.*

Functions

- bool [operator==](#) (const [EntityDescriptor](#) &lhs, const [EntityDescriptor](#) &rhs)

7.6.1 Enumeration Type Documentation

7.6.1.1 BlockValidity

```
enum BlockValidity
```

Enum to describe the status of a block within a validity matrix of an object.

Enumerator

INVALID	
REQUESTED	
VALID	
SHARED	

7.6.1.2 MessageType

```
enum MessageType
```

Enum to describe the different protocol message types.

Enumerator

UNKNOWN	
REQUEST	
REPLY	
ANNOUNCEMENT	

7.6.2 Function Documentation**7.6.2.1 operator==()**

```
bool operator== (
    const EntityDescriptor & lhs,
    const EntityDescriptor & rhs ) [inline]
```

7.7 include/adom/safe_queue.hpp File Reference

```
#include <queue>
#include <mutex>
#include <condition_variable>
#include <adom/parameters.h>
```

Classes

- class [SafeQueue< T >](#)
A thread safe queue.

7.8 include/adom/socket_endpoint.hpp File Reference

```
#include <string>
#include <chrono>
```

Classes

- struct [socket_endpoint](#)

Describes the parameters of an boost asio endpoint. Used for passing endpoints to functions.

7.9 include/adom/translation.h File Reference

```
#include <unistd.h>
#include <string.h>
#include <string>
#include <iostream>
#include <vector>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <adom/parameters.h>
```

Classes

- struct [Structure](#)

To decompose the object sample into data blocks, that are beneficial to the data type or user application's use of the data, the user should specify the proper structure of the decomposed object sample.

Enumerations

- enum [StructureType](#) { [NOT_SPECIFIED](#) = 0 , [ONE_DIMENSIONAL](#) = 1 , [TWO_DIMENSIONAL](#) = 2 }

Enum to describe different types of decomposition strategies for objects samples.

Functions

- `std::vector< unsigned char > translate_from_uchar (unsigned char *data, Structure structure)`
*This function takes data in form of unsigned char *data, e.g. unsigned char array, and translates in into the data format required to construct a cachable Object Sample.*
- `int translate_uchar_access (const unsigned char *pixel_address, const unsigned char *original_data, Structure structure)`
- `unsigned char * translate_to_uchar (std::vector< unsigned char > object_sample_data, Structure structure)`
- `void translate_to_uchar (std::vector< unsigned char > object_sample_data, unsigned char *output_data, Structure structure)`

7.9.1 Enumeration Type Documentation

7.9.1.1 StructureType

```
enum StructureType
```

Enum to describe different types of decomposition strategies for objects samples.

Enumerator

NOT_SPECIFIED	
ONE_DIMENSIONAL	
TWO_DIMENSIONAL	

7.9.2 Function Documentation

7.9.2.1 translate_from_uchar()

```
std::vector<unsigned char> translate_from_uchar (
    unsigned char * data,
    Structure structure ) [inline]
```

This function takes data in form of unsigned char *data, e.g. unsigned char array, and translates in into the data format required to construct a cachable Object Sample.

Todo o check for allignement issues, is the new data buffer big enough etc

7.9.2.2 translate_to_uchar() [1/2]

```
unsigned char* translate_to_uchar (
    std::vector< unsigned char > object_sample_data,
    Structure structure ) [inline]
```

7.9.2.3 translate_to_uchar() [2/2]

```
void translate_to_uchar (
    std::vector< unsigned char > object_sample_data,
    unsigned char * output_data,
    Structure structure ) [inline]
```

7.9.2.4 translate_uchar_access()

```
int translate_uchar_access (
    const unsigned char * pixel_address,
    const unsigned char * original_data,
    Structure structure ) [inline]
```


7.10 src/cpp/adom/application_interface.cpp File Reference

```
#include <adom/application_interface.hpp>
#include <adom/opencv_helper.h>
#include <boost/log/trivial.hpp>
#include <boost/log/core.hpp>
#include <boost/log/expressions.hpp>
#include <boost/log/sources/severity_logger.hpp>
#include <boost/log/sources/record_ostream.hpp>
#include <boost/log/utility/setup/file.hpp>
#include <boost/log/utility/setup/common_attributes.hpp>
#include <boost/log/utility/setup/console.hpp>
```

7.11 src/cpp/adom/directory.cpp File Reference

```
#include <adom/directory.hpp>
#include <boost/log/trivial.hpp>
#include <boost/log/core.hpp>
#include <boost/log/expressions.hpp>
#include <boost/log/sources/severity_logger.hpp>
#include <boost/log/sources/record_ostream.hpp>
#include <boost/log/utility/setup/file.hpp>
#include <boost/log/utility/setup/common_attributes.hpp>
#include <boost/log/utility/setup/console.hpp>
#include <chrono>
```

7.12 src/cpp/adom/log.cpp File Reference

```
#include <adom/log.hpp>
#include <boost/date_time.hpp>
```

7.13 src/cpp/adom/protocol.cpp File Reference

```
#include <adom/protocol.hpp>
```


Index

- ~SafeQueue
 - SafeQueue< T >, [42](#)
- access_end_address
 - AppRequest, [17](#)
- access_start_address
 - AppRequest, [17](#)
- addNewObjectSample
 - Directory, [31](#)
- adom, [9](#)
 - init_console_log, [9](#)
 - init_file_log, [9](#)
- adom_message_length_
 - AdomMessage, [13](#)
- adom_message_overhead
 - parameters.h, [56](#)
- AdomMessage, [11](#)
 - adom_message_length_, [13](#)
 - AdomMessage, [11](#), [12](#)
 - adomMessageToNet, [12](#)
 - clear, [12](#)
 - netToAdomMessage, [12](#)
 - payload_, [13](#)
 - payload_length_, [13](#)
 - type_, [13](#)
- adomMessageToNet
 - AdomMessage, [12](#)
- ANNOUNCEMENT
 - protocol.hpp, [60](#)
- ApplicationInterface, [13](#)
 - ApplicationInterface, [14](#)
 - initialize, [14](#)
 - readPartialData, [14](#), [15](#)
 - registerNewData, [15](#)
 - registerNewTopic, [16](#)
 - stop, [16](#)
- AppLog
 - log.hpp, [51](#)
- AppRequest, [17](#)
 - access_end_address, [17](#)
 - access_start_address, [17](#)
 - object_address, [17](#)
 - sequence_nr, [17](#)
 - topic, [17](#)
- associated_object
 - DataBlockHeader, [21](#)
- associated_object_
 - DataAnnouncement, [20](#)
 - DataTransport, [28](#)
- associated_request_id_
 - DataTransport, [28](#)
- block_cols
 - Structure, [47](#)
- block_id
 - DataBlockHeader, [22](#)
- block_id_
 - DataTransport, [28](#)
- block_rows
 - Structure, [47](#)
- block_size
 - DataBlockHeader, [22](#)
- block_size_
 - DataTransport, [28](#)
- BlockValidity
 - protocol.hpp, [59](#)
- CHANNEL_NUMBER
 - parameters.h, [56](#)
- checkAvailability
 - Directory, [31](#)
- clear
 - AdomMessage, [12](#)
 - DataAnnouncement, [19](#)
 - DataRequest, [24](#)
 - DataTransport, [27](#)
- cv
 - RequestTracking, [40](#)
- data_
 - DataTransport, [29](#)
- data_address_offset
 - DataBlockHeader, [22](#)
- DataAnnouncement, [18](#)
 - associated_object_, [20](#)
 - clear, [19](#)
 - DataAnnouncement, [18](#), [19](#)
 - dataAnnouncementToNet, [19](#)
 - home_directory_, [20](#)
 - message_length_, [20](#)
 - netToDataAnnouncement, [19](#)
 - print, [20](#)
 - structure_, [20](#)
- dataAnnouncementToNet
 - DataAnnouncement, [19](#)
- DataBlockHeader, [21](#)
 - associated_object, [21](#)
 - block_id, [22](#)
 - block_size, [22](#)
 - data_address_offset, [22](#)

- DataBlockHeader, 21
- DataObjectInstance, 22
 - DataObjectInstance, 23
 - object_sequence_nr, 23
 - object_type, 23
- DataRequest, 23
 - clear, 24
 - DataRequest, 24
 - dataRequestToNet, 24
 - home_directory_, 25
 - message_length_, 25
 - netToDataRequest, 25
 - object_block_count_, 25
 - object_block_validity_, 25
 - object_instance_, 26
 - print, 25
 - reader_application_, 26
 - request_id_, 26
- dataRequestToNet
 - DataRequest, 24
- DataTransport, 26
 - associated_object_, 28
 - associated_request_id_, 28
 - block_id_, 28
 - block_size_, 28
 - clear, 27
 - data_, 29
 - DataTransport, 27
 - dataTransportToNet, 27
 - message_length_, 29
 - netToDataTransport, 28
 - print, 28
- dataTransportToNet
 - DataTransport, 27
- dequeue
 - SafeQueue< T >, 42, 43
- Directory, 29
 - addNewObjectSample, 31
 - checkAvailability, 31
 - Directory, 30
 - freeAllObjects, 31
 - getLastObjectSample, 31
 - getLatestSequencenumberOfTrackedTopic, 33
 - getNumberOfTrackedObjects, 33
 - getSample, 33
 - initiate, 34
 - join, 34
 - other_directory_descriptor_, 36
 - printTrackedObjectSamples, 34
 - request_tracking_, 36
 - request_tracking_lock_, 36
 - sendDataRequest, 34
 - setRequestedBlocks, 35
 - stop, 35
 - terminate, 35
- empty
 - SafeQueue< T >, 43
- EMPTY_QUEUE_ERROR
- parameters.h, 56
- enqueue
 - SafeQueue< T >, 43
- entity_id
 - EntityDescriptor, 37
- EntityDescriptor, 36
 - entity_id, 37
 - EntityDescriptor, 37
 - ingress_port, 37
 - ip_address, 37
- findBlock
 - opencv_helper.h, 52
- findBlockFromAddress
 - opencv_helper.h, 53
- findBlocksForReadAccess
 - opencv_helper.h, 53
- freeAllObjects
 - Directory, 31
- FULL_HD_BLOCK_ROWS
 - parameters.h, 56
- FULL_HD_BLOCKS_IN_ROW
 - parameters.h, 56
- FULL_HD_H_PIXEL_PER_IMAGE
 - parameters.h, 56
- FULL_HD_V_PIXEL_PER_IMAGE
 - parameters.h, 56
- getBlockCount
 - ObjectSample, 38
- getHeader
 - ObjectSample, 39
- getHomeDirectory
 - ObjectSample, 39
- getLastObjectSample
 - Directory, 31
- getLatestSequencenumberOfTrackedTopic
 - Directory, 33
- getNumberOfTrackedObjects
 - Directory, 33
- getSample
 - Directory, 33
- getSequenceNumber
 - ObjectSample, 39
- getStructure
 - ObjectSample, 39
- getTopic
 - ObjectSample, 39
- H_PIXEL_PER_BLOCK
 - parameters.h, 57
- HD_BLOCK_ROWS
 - parameters.h, 57
- HD_BLOCKS_IN_ROW
 - parameters.h, 57
- HD_H_PIXEL_PER_IMAGE
 - parameters.h, 57
- HD_V_PIXEL_PER_IMAGE
 - parameters.h, 57

- home_dir_ingress_port
 - parameters.h, 57
- home_directory_
 - DataAnnouncement, 20
 - DataRequest, 25
- IMAGE
 - parameters.h, 56
- include/adom/application_interface.hpp, 49
- include/adom/directory.hpp, 49
- include/adom/log.hpp, 50
- include/adom/opencv_helper.h, 52
- include/adom/parameters.h, 54
- include/adom/protocol.hpp, 59
- include/adom/safe_queue.hpp, 60
- include/adom/socket_endpoint.hpp, 61
- include/adom/translation.h, 61
- ingress_port
 - EntityDescriptor, 37
- init_console_log
 - adom, 9
- init_file_log
 - adom, 9
- initialize
 - ApplicationInterface, 14
- initiate
 - Directory, 34
- INVALID
 - protocol.hpp, 60
- ip_addr
 - socket_endpoint, 45
- ip_address
 - EntityDescriptor, 37
- isAvailable
 - ObjectSample, 39
- join
 - Directory, 34
- log.hpp
 - AppLog, 51
 - logDebug, 51
 - logError, 51
 - logFatal, 51
 - logInfo, 51
 - logTrace, 51
 - logWarning, 51
- log_directory
 - parameters.h, 57
- logDebug
 - log.hpp, 51
- logError
 - log.hpp, 51
- logFatal
 - log.hpp, 51
- logInfo
 - log.hpp, 51
- logTrace
 - log.hpp, 51
- logWarning
 - log.hpp, 51
- makeAvailable
 - ObjectSample, 40
- max_buffer_length
 - parameters.h, 57
- MAX_COUNT_MANAGED_OBJECTS_PER_TOPIC
 - parameters.h, 55
- MAX_NUM_TRACKED_SAMPLES
 - parameters.h, 58
- message_length_
 - DataAnnouncement, 20
 - DataRequest, 25
 - DataTransport, 29
- MessageType
 - protocol.hpp, 60
- netToAdomMessage
 - AdomMessage, 12
- netToDataAnnouncement
 - DataAnnouncement, 19
- netToDataRequest
 - DataRequest, 25
- netToDataTransport
 - DataTransport, 28
- NONE
 - parameters.h, 56
- NOT_SPECIFIED
 - translation.h, 62
- object_address
 - AppRequest, 17
- object_block_count_
 - DataRequest, 25
- object_block_validity_
 - DataRequest, 25
- object_channels
 - Structure, 47
- object_height
 - Structure, 47
- object_instance_
 - DataRequest, 26
- object_sample_data_
 - ObjectSample, 40
- object_sequence_nr
 - DataObjectInstance, 23
- object_type
 - DataObjectInstance, 23
- object_width
 - Structure, 47
- ObjectSample, 38
 - getBlockCount, 38
 - getHeader, 39
 - getHomeDirectory, 39
 - getSequenceNumber, 39
 - getStructure, 39
 - getTopic, 39
 - isAvailable, 39

- makeAvailable, 40
- object_sample_data_, 40
- ObjectSample, 38
- printObjectSample, 40
- ONE_DIMENSIONAL
 - translation.h, 62
- opencv_helper.h
 - findBlock, 52
 - findBlockFromAddress, 53
 - findBlocksForReadAccess, 53
- operator==
 - protocol.hpp, 60
- other_directory_descriptor_
 - Directory, 36
- parameters.h
 - adom_message_overhead, 56
 - CHANNEL_NUMBER, 56
 - EMPTY_QUEUE_ERROR, 56
 - FULL_HD_BLOCK_ROWS, 56
 - FULL_HD_BLOCKS_IN_ROW, 56
 - FULL_HD_H_PIXEL_PER_IMAGE, 56
 - FULL_HD_V_PIXEL_PER_IMAGE, 56
 - H_PIXEL_PER_BLOCK, 57
 - HD_BLOCK_ROWS, 57
 - HD_BLOCKS_IN_ROW, 57
 - HD_H_PIXEL_PER_IMAGE, 57
 - HD_V_PIXEL_PER_IMAGE, 57
 - home_dir_ingress_port, 57
 - IMAGE, 56
 - log_directory, 57
 - max_buffer_length, 57
 - MAX_COUNT_MANAGED_OBJECTS_PER_TOPIC, 55
 - MAX_NUM_TRACKED_SAMPLES, 58
 - NONE, 56
 - PRINTF_BINARY_PATTERN_INT8, 55
 - PRINTF_BYTE_TO_BINARY_INT8, 55
 - PROTOCOL_TEST, 55
 - publisher_ip, 58
 - req_dir_ingress_port, 58
 - string_length, 58
 - subscriber_ip, 58
 - Topic, 55
 - TOTAL_BLOCK_SIZE, 58
 - V_PIXEL_PER_BLOCK, 58
- payload_
 - AdomMessage, 13
- payload_length_
 - AdomMessage, 13
- port
 - socket_endpoint, 45
- print
 - DataAnnouncement, 20
 - DataRequest, 25
 - DataTransport, 28
- PRINTF_BINARY_PATTERN_INT8
 - parameters.h, 55
- PRINTF_BYTE_TO_BINARY_INT8
 - parameters.h, 55
- printObjectSample
 - ObjectSample, 40
- printTrackedObjectSamples
 - Directory, 34
- protocol.hpp
 - ANNOUNCEMENT, 60
 - BlockValidity, 59
 - INVALID, 60
 - MessageType, 60
 - operator==, 60
 - REPLY, 60
 - REQUEST, 60
 - REQUESTED, 60
 - SHARED, 60
 - UNKNOWN, 60
 - VALID, 60
- PROTOCOL_TEST
 - parameters.h, 55
- publisher_ip
 - parameters.h, 58
- queue_event
 - SafeQueue< T >, 44
- queue_lock
 - SafeQueue< T >, 44
- reader_application_
 - DataRequest, 26
- readPartialData
 - ApplicationInterface, 14, 15
- received_blocks
 - RequestTracking, 41
- registerNewData
 - ApplicationInterface, 15
- registerNewTopic
 - ApplicationInterface, 16
- REPLY
 - protocol.hpp, 60
- req_dir_ingress_port
 - parameters.h, 58
- REQUEST
 - protocol.hpp, 60
- request_id_
 - DataRequest, 26
- request_tracking_
 - Directory, 36
- request_tracking_lock_
 - Directory, 36
- REQUESTED
 - protocol.hpp, 60
- requested_blocks
 - RequestTracking, 41
- RequestTracking, 40
 - cv, 40
 - received_blocks, 41
 - requested_blocks, 41
- safe_queue

- SafeQueue< T >, 44
- SafeQueue
 - SafeQueue< T >, 42
- SafeQueue< T >, 41
 - ~SafeQueue, 42
 - dequeue, 42, 43
 - empty, 43
 - enqueue, 43
 - queue_event, 44
 - queue_lock, 44
 - safe_queue, 44
 - SafeQueue, 42
- sendDataRequest
 - Directory, 34
- sequence_nr
 - AppRequest, 17
- setRequestedBlocks
 - Directory, 35
- SHARED
 - protocol.hpp, 60
- socket_endpoint, 45
 - ip_addr, 45
 - port, 45
 - socket_endpoint, 45
- src/cpp/adom/application_interface.cpp, 63
- src/cpp/adom/directory.cpp, 63
- src/cpp/adom/log.cpp, 63
- src/cpp/adom/protocol.cpp, 63
- stop
 - ApplicationInterface, 16
 - Directory, 35
- string_length
 - parameters.h, 58
- Structure, 46
 - block_cols, 47
 - block_rows, 47
 - object_channels, 47
 - object_height, 47
 - object_width, 47
 - Structure, 46, 47
 - type, 47
- structure_
 - DataAnnouncement, 20
- StructureType
 - translation.h, 61
- subscriber_ip
 - parameters.h, 58
- terminate
 - Directory, 35
- Topic
 - parameters.h, 55
- topic
 - AppRequest, 17
- TOTAL_BLOCK_SIZE
 - parameters.h, 58
- translate_from_uchar
 - translation.h, 62
- translate_to_uchar
 - translation.h, 62
- translation.h, 62
- translate_uchar_access
 - translation.h, 62
- translation.h
 - NOT_SPECIFIED, 62
 - ONE_DIMENSIONAL, 62
 - StructureType, 61
 - translate_from_uchar, 62
 - translate_to_uchar, 62
 - translate_uchar_access, 62
 - TWO_DIMENSIONAL, 62
- TWO_DIMENSIONAL
 - translation.h, 62
- type
 - Structure, 47
- type_
 - AdomMessage, 13
- UNKNOWN
 - protocol.hpp, 60
- V_PIXEL_PER_BLOCK
 - parameters.h, 58
- VALID
 - protocol.hpp, 60